# Time Optimal Trajectory Planning in Dynamic Environments

Paolo Fiorini* and Zvi Shiller†

* Jet Propulsion Laboratory
California Institute of Technology
Pasadena, CA 91109 USA

† Department of Mechanical, Nuclear and Aerospace Engineering
University of California, Los Angeles
Los Angeles, CA 90024 USA

## Abstract

This paper presents a direct method for computing the time optimal trajectory for a robot among stationary and moving obstacles, subject to robot's dynamics and actuator constraints. The motion planning problem is first formulated as an optimization problem, and then solved numerically using a gradient descent. The initial guess for the optimization is generated using a method based on the concept of Velocity Obstacles. The method is demonstrated for a 2-DOF planar manipulator moving in static and dynamic environments.

## 1. Introduction

Motion planning is central to the operation of autonomous robots. It concerns the generation of a trajectory from a start to goal that satisfies the tasks objectives, such as minimizing path distance or motion time, while avoiding obstacles in the environment and satisfying the robot mechanics (kinematics and dynamics). We distinguish between *planning* and *control* in that the former generates a nominal trajectory, whereas the latter tracks that trajectory. Robot motion planning is generally too complex to be handled by on-line feedback controllers due to the nonlinear state constrains introduced by the obstacles and the highly nonlinear and coupled nature of robot mechanics.

Traditionally, motion planning has been treated as a *kinematic* problem, i.e. determining the path that avoids obstacles without concern to robot speeds. This was first extensively addressed for articulated robots by transforming the problem into the *configuration space*, in which the robot reduces to a point and the obstacles map into C-space obstacles [26,30]. The focus in this body of work has centered on computational complexity and completeness (the ability of the algorithm to find a path if one exists). More recently the *kinematic* problem was extended to car-like robots, which are subject to *non-holonomic* kinematic constraints due to the assumption of no slip between the wheels and ground. Here the focus has centered on obstacle avoidance [28] and on minimizing path distance [27].

While solving a problem fundamental to robotics, *kinematic* motion planning ignores the important effects of robot dynamics which become significant at all but the lowest speeds. For example, non-holonomic motion planning of a car is useful for parking [32], which is usually

done at very low speeds, but is all but meaningless for high speed emergency maneuvers [38]. Similarly, obstacle-free paths computed using robot kinematics may be dynamically infeasible at even moderate speeds, causing the robot to deviate from the *kinematic* path due to its dynamics and limited actuator efforts. This gave rise to *dynamic* motion planning[1], which produces a *trajectory* in the state space rather than just a *path* in the configuration space. Planning in the sate space, while computationally more extensive, allows one to minimize *dynamic* cost functions, such as time or energy. These problems have been treated previously for both articulate] [35,36] and mobile robots [37].

We distinguish between motion planning in *static* and in *dynamic* environments. In static environments, the obstacles are static, and the robot is the only one that moves, whereas in dynamic environments, both robot and obstacles move. Typical examples of dynamic environments include manufacturing tasks in which robot manipulators track and retrieve parts from moving conveyers, and intelligent vehicles negotiating freeway traffic.

. Motion planning in dynamic environments was originally addressed by adding the time dimension to the robot's configuration space, assuming bounded velocity and known trajectories of the obstacles [9,19,33]. Reif and Sharir [33] solved the planar problem for a polygonal robot among many moving polygonal obstacles, by searching a visibility graph in the configuration-time space. Erdmann and Lozano-Pérez [9] discretized the configuration-time space to result in a sequence of configuration space slices at successive time intervals. This method essentially solves the static planning problem at every slice and joins adjacent solutions. Fujimura and Samet [19] used a cell decomposition to represent the configuration-time space, and joined empty cells to connect start to goal.

Another approach to dynamic motion planning is to decompose the problem into smaller problems: path planning and velocity planning. This method first computes a feasible path among the static obstacles, and represents it as a parametric curve in the arc length. Then, the intersections of the moving obstacles with the path are represented as forbidden regions in an arc length-time plane. The velocity along the path is chosen to avoid the forbidden regions [15,16,18,20,25,29]. Kant and Zucker [25] selected both path and velocity profile using a visibility graph approach. Lee and Lee [29] developed independently a similar approach for two cooperating robots, and compared the effects of delay and velocity reduction on motion time. Fraichard [15] considered acceleration bounds, and used a search in a state-t)irrie space $(s, \dot{s}, t)$ to compute the velocity profile yielding a minimum-time trajectory. Fraichard and Laugier [16] considered adjacent paths that could be reached from the nominal path when the nominal path becomes blocked by a moving obstacle. Fujimura [18] considered the case of a robot moving on a fixed time-dependent network, whose nodes could be temporarily occluded by moving obstacles.

A different approach consists of generating the accessibility graph of the environment, which is an extension of the visibility graph [20,21]. Fujimura and Samet [20] defined it as the locus of points on the obstacles which are reachable by the robot moving at maximum speed. These points form the *collision front*, and can be linked together to construct a path from start to goal. The accessibility graph has the property that, if the robot moves faster than the obstacles, the path computed by searching the graph is time-minimal. This concept was extended in [17] to the case of slowly moving robots and transient obstacles, i.e. obstacles that could appear and disappear in the environment.

None of the previous methods considered the non-linear robot dynamics, and none produced time optimal motions. Time-optimal motions have obvious benefits in industrial applications by reducing cycle times and thus increasing the productivity of automated manufacturing systems.

[1] Others use *dynamic* motion planning to denote motion planning in *dynamic* environments [26], which is a subset of our definition.

Other application domains, such as intelligent vehicles and air traffic control, may benefit from time-optimal motions by minimizing the recovery time from emergency situations and when defining emergency maneuvers.

The time-optimal motion planning problem in static environments has been treated previously, beginning with the work by Kahn and Roth [24], who solved the problem for a linearized robot model, using the Pontryagin's Minimum Principle (JIM]'). The full robot model was used in [31], assuming bang-bang control and using a steepest decent over the switching times, derived to satisfy the necessary conditions of optimality stated by the PMP. However, the most efficient methods to date seem to consist of parameter optimizations over the trajectory [2,23,36] which are similar to the 1 Differential Inclusions introduced in [34], and the Inverse Dynamic Optimization introduced by 1 Bryson [4].

In this paper, we present a method for computing the time optimal trajectories of a robot moving in a dynamic environment. To make the problem computationally tractable, we restrict the treatment to the plane and assume circular robot and obstacles. We also assume a full knowledge of the environment.

Central to this approach is the computation of the initial guess for the optimization. This is done by utilizing the concept of Velocity obstacle [10,1 1], which maps the dynamic environment into the robot velocity space. The velocity obstacle is the first-order approximation of the robot's velocities that would cause a collision with an obstacle at some future time, within a given time horizon. Feasible avoidance maneuvers are computed simply selecting velocities outside the velocity obstacle, and satisfying additional velocity cxnlst)-silks computed from robot dynamics and actuator constraints. The in it ial guess of the optimal trajectory is computed by a global search over a tree of feasible avoidance maneuvers generated at discrete time intervals so as to minimize time to the goal.

The optimal trajectory is computed using a steepest descent algorithm over the admissible controls [G- 8], modified to consider time varying state inequality constraints. The state inequality constraints due to the moving obstacles are considered by transforming them into state-dependent control constraints. The method was implemented for intelligent vehicles negotiating freeway traffic [38], and for a planar SCARA robot, considering its full nonlinear dynamics and moving circular obstacles [10]. Examples of the latter are presented in this paper.

The paper is organized as follows. Section 2 formulates the motion planning problem as a minimum time problem and presents the numerical method for computing the optimal solution satisfying state inequality constraints and state-dependent control constraints. Then, Section 3 addresses the problem of generating the nominal trajectory for the numerical optimization. Finally, examples of optimal trajectories of a SCARA robot avoiding fixed and moving obstacles are presented in Section 4.

## 2. The Dynamic Optimization

The dynamic motion planning problem in the context of this paper consists of determining the trajectory between two specified boundary conditions that avoids all static and moving obstacles and minimizes motion time. This is formulated as an optimization problem with time-varying state constraints, and is solved numerically using the steepest descent method [8], as discussed next.

## 2.1. Problem Formulation

The motion planning problem can be formulated as follows: Find the control $\mathbf{u}^*(t) \in \mathbf{U}$ in $t_0 \leq t \leq t_f$, which minimizes the cost function $J$:

$$\min_{u(t)\in \mathbf{U}} J = \min_{u(t)\in \mathbf{U}} \int_{t_0}^{t_f} L(\mathbf{x}(t), \mathbf{u}(t))dt = \phi_{min}(\mathbf{x}(t_f), t_f) = t_f \tag{1}$$

where $t_f$ is free, subject to robot dynamics

$$x = .7(\ X,11) = f(x) - 1\ g(\mathbf{x})\mathbf{u} \tag{2}$$

admissible controls

$$\mathbf{U} = \{\mathbf{u} \mid \mathbf{u}_{min} \leq \mathbf{u} \leq \mathbf{u}_{max}\} \tag{3}$$

initial conditions

$$\mathbf{x}(t_0) = \mathbf{x}_0 \tag{4}$$

terminal manifold

$$\Omega(\mathbf{x}(t_f), t_f) = o \tag{5}$$

and state inequality constraints due to the moving obstacles:

$$\Psi : \bigcup_{i=1}^{n} [S_i(\mathbf{x}(t), t) \geq 0] \tag{6}$$

where $S_i(\mathbf{x}(t), t)$ represents the time-varying boundaries of the moving obstacles.

The original problem calls for a fixed final point. 1 lowever, we assume instead a terminal manifold (a hyper-sphere around the final point) so that we can use influence functions to compute the initial condition of the 1 agrange multipliers, and thus avoid using the more sensitive shooting method [3].

State inequality constraints are generally difficult to satisfy although necessary conditions for optimality nave been developed for such problems [22,39]. One way to consider state inequality constraints is to transform them into state-dependent control equality constraints, active only when the robot slides along the obstacle boundary [7,8].

To demonstrate the treatment of the state inequality constraints, we consider the single obstacle:

$$\Psi : S(\mathbf{x}(t), t) \geq 0\ ,\ S(x) \in \Re^m \tag{7}$$

where $m$ is the dimension of the position space. Differentiating (7) with respect to time $p$ times until it becomes explicit in the control $\mathbf{u}$, and assuming an active constraint, yields the state-dependent control constraint

$$S^{(p)}(\mathbf{x}, \mathbf{u}) = 0 \tag{8}$$

where $S^{(p)}$ denotes the $pth$ derivative of $S$, with $p$ being the order of the constraint.

A solution satisfying (8) dots not necessarily satisfy (7), unless it passes through at least one point satisfyin g (7) and all the derivatives or order less then $p$. We choose this point to be the initial entry point of the constrained arc, at time $t_1 > t_0$. The inequality constraint (7) is

thus replaced by the tangency point condition, $\Psi_1$, and the state-dependent equality constraint, $\Psi_2$:

$$\Psi_1 \; : \; \left| \begin{array}{l} S(\mathbf{x}(t_1), t_1) = 0 \\ \dot{S}(\mathbf{x}(t_1), t_1) = 0 \\ \\ S^{(p-1)}(\mathbf{x}(t_1), t_1) = 0 \end{array} \right| \tag{9}$$

$$\Psi_2 \; : \; S^{(p)}(\mathbf{x}(t), \mathbf{u}(t), t) = 0 \qquad t_1 \leq t \leq t_2 \tag{10}$$

where $t_1$ is the entry time, and $t_2$ is the exit time of the constrained arc. This also modifies the admissible controls (3) to:

$$\mathcal{U} : \left\{ \begin{array}{l} \mathbf{U} : \mathbf{u}_{min} \leq \mathbf{u} \leq \mathbf{u}_{max} \\ S^{(p)}(\mathbf{x}(t), \mathbf{u}(t), t) = 0 \quad \text{for} \quad t \in (t_1, t_2] \end{array} \right. \tag{11}$$

The addition of the tangency constraint, $\Psi_1$, thus transforms the original Two Point Boundary Value Problem (1) into a Three Point Boundary Value Problem (for a single moving obstacle), which is solved numerically using the method discussed next.

Note that this treatment of the state inequality constraints may over-constrain the problem since the trajectory is forced to satisfy the static constraint as an equality along a finite arc. Consequently, this approach cannot find solutions that touch the state constraint at multiple isolated points [22]. This, however, has been shown to affect only constraints of order higher than two, and is hence not an issue for the circular obstacles treated here [22].

### 2.2. Numerical Computation

We apply the steepest descent method, which rigorously satisfies a set of necessary optimality conditions. This method was originally developed in [6], modified to include state dependent control inequality constraints in [8], and modified to consider bang-bang controls in [31].

The steepest descent method iteratively computes the optimal controls by following the negative gradient of the augmented cost function with respect to the controls and the final time. The gradient is derived by adjoining the differential of the cost function with the differentials of the terminal manifold and the tangency-point constraint, as discussed below.

#### 2.2.1. The Differential of the Performance Index

Following the classic approach to constrained optimization [7], system dynamics and control constraints are adjoined to the performance index $J$ using two arrays of Lagrange functions $\lambda_\phi(t) \in \Re^n$ and $\mu(t) \in \Re^k$, where $n$ is the dimension of the state space, and $k$ is the number of active state-dependent control constraints. This leads to the performance index $\tilde{J}$:

$$\tilde{J} = \phi(\mathbf{x}(t_f)) + \int_{t_0}^{t_f} \left[ \lambda_\phi^T (\mathcal{F}(\mathbf{x}, \mathbf{u}) - \dot{\mathbf{x}}) + \mu^T \varphi(\mathbf{x}, \mathbf{u}) \right] dt \tag{12}$$

where

$$\varphi = \left\{ \begin{array}{ll} 0 & t \in [t_1, t_2] \\ S^{(p)} & t \in (t_1, t_2) \end{array} \right. \tag{13}$$

and $\mu$ is a vector of Kuhn-Tucker multipliers [7]

$$\mu = \left\{ \begin{array}{ll} 0 & \text{when the constraint is inactive} \\ -\mathbf{\Lambda}^T \mathbf{g}(\mathbf{x}) \left( \frac{\partial \varphi(\mathbf{x}, \mathbf{u})}{\partial \mathbf{u}} \right)^{-1} & \text{otherwise} \end{array} \right. \tag{14}$$

By defining the Hamiltonian as:

$$\mathcal{H}(\lambda_\phi, \mathbf{x}, \mathbf{u}) = \lambda_\phi^T \mathcal{F}(\mathbf{x}, \mathbf{u}) + \mu^T \varphi(\mathbf{x}, \mathbf{u}) \tag{15}$$

and by choosing:

$$\dot{\lambda}_\phi(t) = -\left(\frac{\partial \mathcal{H}}{\partial \mathbf{x}}\right)^T \tag{16}$$

$$\lambda_\phi(t_f) = \left(\frac{\partial \phi}{\partial \mathbf{x}}\right)_{t_f} \tag{17}$$

we reduce the differential $d\widetilde{J}$ to:

$$d\widetilde{J} = \int_{t_0}^{t_f} \frac{\partial \mathcal{H}}{\partial \mathbf{u}} \delta \mathbf{u} d\tau + \left(\frac{\partial \phi}{\partial t} + \mathcal{H}\right)_{t_f} dt_f \tag{18}$$

This establishes the relations between variations in the independent variables, $\mathbf{u}$ and $t_f$, and variations in the cost function for the unconstrained problem.

### 2.2.2. The Differential of the Terminal Constraint

The differential of the terminal constraint $\Omega$ is:

$$(d\Omega)_{t_f} = \left(\frac{\partial \Omega}{\partial x} dx + \frac{\partial \Omega}{\partial t} dt\right)_{t_f} \tag{19}$$

Following the derivation in Appendix A, and choosing multipliers $\lambda_\Omega \in \Re^n \times \Re^l$ (where $l$ is the number of terminal constraints) to satisfy:

$$\dot{\lambda}_\Omega(t) = -\left(\frac{\partial \mathcal{F}}{\partial x}\right)^T \lambda_\Omega(t) \tag{20}$$

$$\lambda_\Omega(t_f) = \left(\frac{\partial \Omega}{\partial x}\right)_{t_f} \tag{21}$$

the differential $d\Omega$ (19) reduces to:

$$(d\Omega)_{t_f} = \int_{t_0}^{t_f} \lambda_\Omega^T \frac{\partial \mathcal{F}}{\partial \mathbf{u}} \delta \mathbf{u} d\tau + \left(\frac{\partial \Omega}{\partial x} \dot{x} + \frac{\partial \Omega}{\partial t}\right)_{t_f} dt_f \tag{22}$$

### 2.2.3. The Differential of the Point Constraint

Similarly, the differential of the intermediate tangency constraints, $\Psi_1$, at time $t_1$, is:

$$(d\Psi_1)_{t_1} = \left(\frac{\partial \Psi_1}{\partial x} dx + \frac{\partial \Psi_1}{\partial t} dt\right)_{t_1} \tag{23}$$

Following the derivation in the Appendix A, and choosing the Lagrange functions $\lambda_\Psi \in \Re^n \times \Re^k$ (where $k$ is the number of constraints $\Psi_1$)

$$\dot{\lambda}_\Psi(t) = -\left(\frac{\partial \mathcal{F}}{\partial x}\right)^T \lambda_\Psi(t) \tag{24}$$

$$\lambda_\Psi(t_1) = \left(\frac{\partial \Psi}{\partial x}\right)_{t_1} \tag{25}$$

the differential $d\Psi$ (23) reduces to:

$$(d\Psi_1)_{t_1} = \int_{t_0}^{t_1} \lambda_\Psi^T \frac{\partial \mathcal{F}}{\partial \mathbf{u}} \delta \mathbf{u} d\tau + \left(\frac{\partial \Psi}{\partial x} \dot{x} + \frac{\partial \Psi}{\partial t}\right)_{t_1} dt_1 \tag{26}$$

### 2.2.4. 1 Discontinuity of the Lagrange Functions

The Lagrange functions $\lambda_\phi$, and $\lambda_\Omega$ are integrated through the entry point of the constrained arc at $t_1$, where they are discontinuous. This discontinuity is computed as a function of the jump in the acceleration (for a second order system) across the entry point to the constrained arc [5] (see also Appendix B):

$$\lambda_\phi^T(t_1^-) = \lambda_\phi^T(t_1^+)\left(I - \frac{\dot{\mathbf{x}}(t_1^-) - \dot{\mathbf{x}}(t_1^+)}{S^{(p)}(t_1)}\frac{\partial S^{(p-1)}}{\partial \mathbf{x}}\bigg|_{t_1}\right) \tag{27}$$

$$\lambda_\Omega^T(t_1^-) = \lambda_\Omega^T(t_1^+)\left(I - \frac{\dot{\mathbf{x}}(t_1^-) - \dot{\mathbf{x}}(t_1^+)}{S^{(p)}(t_1)}\frac{\partial S^{(p-1)}}{\partial \mathbf{x}}\bigg|_{t_1}\right) \tag{28}$$

### 2.2.5. The 1 Differential of the Augmented Performance Index

The differential of the augmented performance index $d_2J$ consists of the differentials (22) and (26), appended to the differential $dJ$ with the constant multipliers $\eta$ and $\nu$:

$$\begin{aligned} d_2J = {} & \left[\frac{\partial \phi}{\partial t} + \nu^T\frac{\partial \Omega}{\partial t} + \left(\frac{\partial \phi}{\partial \mathbf{x}} + \nu^T\frac{\partial \Omega}{\partial \mathbf{x}}\right)\dot{\mathbf{x}} + \mu^T\varphi(\mathbf{x},\mathbf{u})\right]_{t_f} dt_f \\ & + \int_{t_0}^{t_1^-}\left[\frac{(\lambda_\phi^T + \nu^T\lambda_\Omega^T + \eta^T\lambda_\Psi^T)\mathcal{F} + \mu^T\varphi(\mathbf{x},\mathbf{u})}{\partial \mathbf{u}}\right]\delta\mathbf{u}d\tau \\ & + \int_{t_1^+}^{t_f}\left[\frac{(\lambda_\phi^T + \nu^T\lambda_\Omega^T)\mathcal{F} + \mu^T\varphi(\mathbf{x},\mathbf{u})}{\partial \mathbf{u}}\right]\delta\mathbf{u}d\tau \end{aligned} \tag{29}$$

Note that the multipliers $\lambda_\Psi$ are defined only between $t_0$ to $t_1$, since $\Psi_1$ is not affected by the states after $t_1$. Setting $\lambda_\Psi(t) = 0$ for $t > t_1$ we can define an augmented Lagrange function, $\mathbf{\Lambda}$:

$$\mathbf{\Lambda}^T = \lambda_\phi^T + \nu^T\lambda_\Omega^T + \eta^T\lambda_\Psi^T \tag{30}$$

which yields the Hamiltonian:

$$\mathcal{H}(\mathbf{\Lambda},\mathbf{x},\mathbf{u}) = \mathbf{\Lambda}^T\mathcal{F}(\mathbf{x},\mathbf{u}) + \mu^T\varphi(\mathbf{x},\mathbf{u}) \tag{31}$$

and reduces (30) to:

$$d_2J = \left(\frac{\partial \phi}{\partial t} + \nu^T\frac{\partial \Omega}{\partial t} + \mathcal{H}\right)_{t_f} dt_f + \int_{t_0}^{t_1^-}\mathcal{H}_u\delta u d\tau + \int_{t_1^+}^{t_f}\mathcal{H}_u\delta u d\tau \tag{32}$$

This establishes the relations between variations in the independent variables, u and $t_f$, and variations in the cost function for the *constrained* problem, including the terminal manifold, the tangency point, and the state-dependent control constraint. Assuming bang-bang control, we use these relations to compute the variations in the switching times that would zero the differential of the augmented cost function.

### 2.2.6. *The* Bang-Bang Solution

It is easy to show that the solution for minimum-time problems consists of bang-bang controls, for systems linear in the controls, excluding singular arcs [7] [40]. By assuming bang-bang control we reduce the functional optimization to a parameter optimization over the switching times. The number of switches is approximated from the initial guess, as discussed later, and the singular arcs are approximated by a finite number of switches [31].

For bang-bang controls, the variations $\delta u_i$ in (32) are replaced with:

$$\delta u_i = (\alpha_m - \alpha_m)\, sgn(dt_{ij}) \tag{33}$$

where $sgn$ is the signum function, and $dt_{ij}$ is the change of the $j$th switching time for control $u_i$. Note that $\delta u_i \neq 0$ only at the switching times where $u_i$ switches between the extremes. Therefore $\delta u_i$ is represented by

$$\delta u_i = (-1)^{j-1} \Delta\alpha\, dt_{ij} \tag{34}$$

where

$$\Delta\alpha = \alpha_M - \alpha_m \tag{35}$$

Using *(34)* we now discretize the augmented cost function $dJ$ of *(32)* as a function of the switching times:

$$\begin{aligned} dJ = & \sum_{i=1}^{m}\sum_{j=1}^{s_{1,i}} (\mathcal{H}_{u_i})_{t_{ij}} \Delta\alpha dt_{ij} + \sum_{i=1}^{m}\sum_{j=1}^{s_{2,i}} (\mathcal{H}_{u_i})_{t_{ij}} \Delta\alpha dt_{ij} \\ & + \sum_{i=1}^{m}\sum_{j=1}^{s_{3,i}} (\mathcal{H}_{u_i})_{t_{ij}} \Delta\alpha dt_{ij} + \left(\frac{\partial\phi}{\partial t} + \nu^T \frac{\partial\Omega}{\partial t} + \mathcal{H}\right)_{t_f} dt_f \end{aligned} \tag{36}$$

where $s_1$ represents the segment of the trajectory before the obstacle, $s_2$ represents the constrained arc, and obstacle, $s_3$ is the segment of the trajectory from the obstacle to the target, and $m$ is the dimension of $\mathbf{u}$. Since the second term in (36) corresponds to the constrained arc, the corrections $dt_{ij}$ are computed only for the controls not determined from $S^{(p)}(\mathbf{x}, \mathbf{u}, t) = 0$.

The objective now is to determine the variations $dt_{ij}$ that would minimize the differential $dJ$. This can be done by following the negative gradient of $dJ$ defined by the coefficients of the $dt_{ij}$ in (36). The step size of each move is determined by adding a quadratic term in $dt_{ij}$ and $dt_f$ to (36) [31]:

$$\begin{aligned} d\hat{J} = & \sum_{i=1}^{m}\sum_{j=1}^{s_{1,i}} \left((\mathcal{H}_{u_i})_{t_{ij}} \Delta\alpha dt_{ij} + \frac{1}{2} w_{ii} \Delta\alpha_i^2 dt_{ij}^2\right) \\ & + \sum_{i=1}^{m}\sum_{j=1}^{s_{2,i}} (\mathcal{H}_{u_i})_{t_{ij}} \Delta\alpha dt_{ij} + \frac{1}{2} w_{ii} \Delta\alpha_i^2 dt_{ij}^2 \\ & + \sum_{i=1}^{m}\sum_{j=1}^{s_{3,i}} (\mathcal{H}_{u_i})_{t_{ij}} \Delta\alpha dt_{ij} + \frac{1}{2} w_{ii} \Delta\alpha_i^2 dt_{ij}^2 \\ & + \left(\frac{\partial\phi}{\partial t} + \nu^T \frac{\partial\Omega}{\partial t} + \mathcal{H}\right)_{t_f} dt_f + \frac{1}{2} b (dt_f)^2 \end{aligned} \tag{37}$$

where $b$ is a positive number, and $w_{ii}$ are the elements of a diagonal positive definite matrix.

The step size that minimizes (37) is given by:

$$dt_{ij} = -\frac{(\mathcal{H}_{u_i})_{t_{ij}}}{w_{ii}\Delta\alpha} \tag{38}$$

$$dt_f = -\frac{1}{b}\left(\mathcal{H} + \frac{\partial\phi}{\partial t} + \nu^T \frac{\partial\Omega}{\partial t}\right)_{t_f} \tag{39}$$

The values of $dt_{ij}$ and $dt_f$ in equations (38) and (39) depend on the multipliers $\eta$ and $\nu$, which are computed by back-substituting (38) and (39) in (22) and (26), and by multiplying

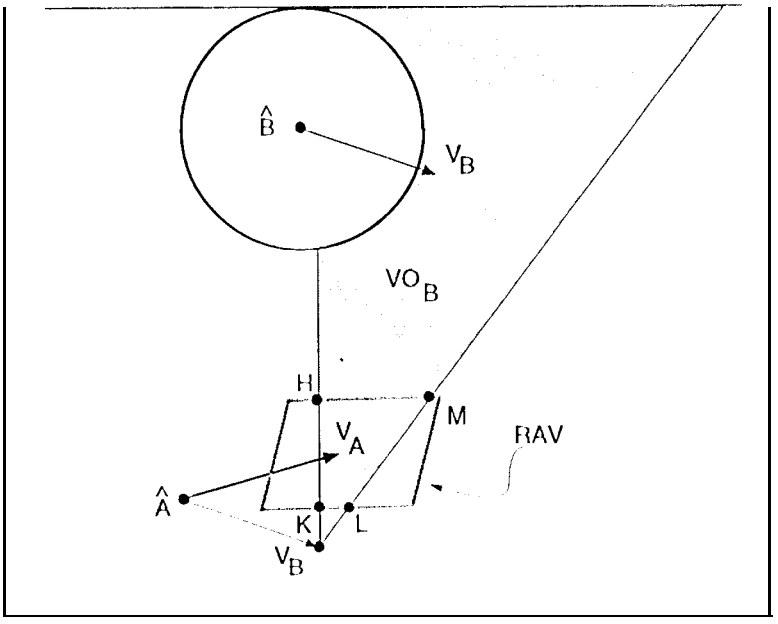


Figure 1: The feasible avoidance velocities RAV

$d\Psi_1(t_1)$ and $d\Omega(t_f)$ by - c, with $\epsilon$ a small positive number. This scales the improvements in $\eta$ and $\nu$ to satisfy the first-order necessary conditions of optimality.

With this substitution, equations (22) and (26) yield:

$$\eta = - I_{\Psi\Psi}^{-1}(- \epsilon\, d\Psi_{t_1} + I_{\Psi\Omega}\nu + I_{\Psi\phi}) \tag{40}$$

$$\nu = - \left( {}^1I_{\Omega\Omega} + {}^1I_{\Omega\Psi}\, {}^1I_{\Psi\Psi}^{-1}\, {}^1I_{\Psi\Omega} + {}^2I_{\Omega\Omega} + {}^3I_{\Omega\Omega} + \frac{1}{b}\left(\frac{d\Omega}{dt}\frac{d\Omega}{dt}^T\right)_{t_f} \right)^{-1}$$

$$\left( - \epsilon\, d\Omega_{t_f} + {}^1I_{\Omega\phi} - {}^1I_{\Omega\Psi}\, {}^1I_{\Psi\Psi}^{-1} \left( - \epsilon\, d\Psi_{t_1} + {}^1I_{\Psi\phi} \right) + {}^2I_{\Omega\phi} + {}^3I_{\Omega\phi} + \frac{1}{b}\left(\frac{d\Omega}{dt}\frac{d\phi}{dt}\right)_{t_f} \right)$$

where the terms ${}^lI_{hk}$ are defined as:

$$ {}^lI_{hk} = \sum_{i=1}^{m} \sum_{j=1}^{s_{l,i}} (\lambda_h^T \frac{\partial \mathcal{F}}{\partial u_i} w_{ii}^{-1} \frac{\partial \mathcal{F}}{\partial u_i}^T \lambda_k)_{t_{ij}} \tag{41}$$

with $h = \Psi_1, \Omega$, $k = \Psi_1, \Omega, \phi$, $l = 1, 2, 3$, representing *before*, *on* and *after* the state! constraint, and $i$ indicating the independent controls.

This procedure reduces the differential defined in (32) to zero, which also satisfies the necessary conditions of optimality stated by the Pontryagin Minimum Principle, as discussed in [10]

## 3. The Initial Guess

The dynamic optimization discussed earlier converges only to a local minimum, which depends on the initial guess. Since the dynamic motion planning problem is generally not convex, i.e. it has multiple local minima, selecting the appropriate initial guess would determine

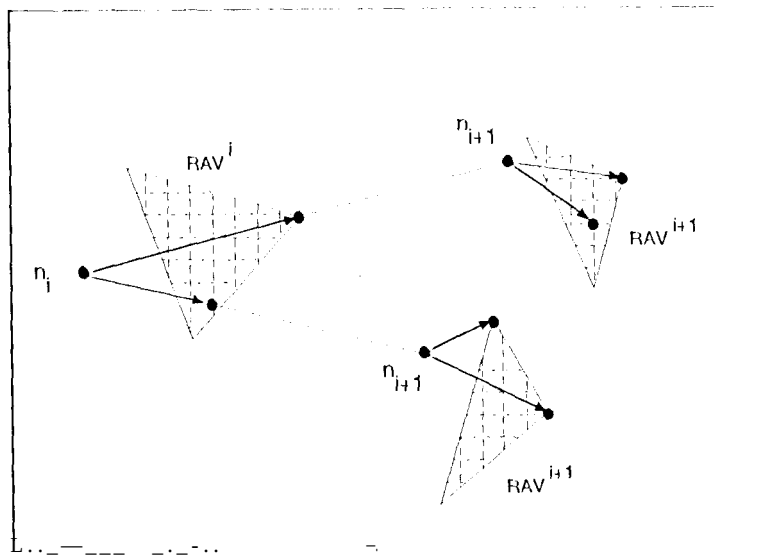


Figure 2: Tree representation for the global search.

the quality of the solution. While it is generally desirable to compute the global minimal trajectory, it is equally important to obtain a trajectory specified in terms of the sequence of avoidance and the side from which each obstacle is being avoided. Selecting an initial guess in dynamic environments is in itself a dynamic motion planning problem, as discussed earlier in the Introduction. Imposing a desired structure makes the problem harder.

An efficient method for solving both problems has been recently developed [13]. It generates trajectories that are both collision-free and dynamically feasible. Below, we first briefly summarize this approach, and then compute a bang-bang approximation for the controls.

### 3.1. Generating the Trajectory

The method for generating feasible trajectories in dynamic environments is based on the concept of velocity obstacles, which is a first-order approximation of the robot velocities that would cause a collision with some obstacle at some future time [10,12]. Collision is avoided by selecting velocities outside the union of the velocity obstacles due to all moving and static obstacles.

To ensure that the selected maneuver is also dynamically feasible, we impose additional velocity constraints due to robot dynamics and actuator constraints, as shown in Figure 1. Figure 1 shows the velocity obstacle of $\hat{B}$, moving at some velocity $\mathbf{v}_B$, with respect to a point robot, $\hat{A}$. Also shown are the feasible velocities RAV, which for a planar robot are represented by a parallelogram. The feasible avoidance velocities are confined to the set defined by the difference between the feasible avoidance velocities and the velocity obstacle.

An avoidance maneuver consists of a velocity vector and a time interval over which that velocity is applied. Maneuvers can be selected to minimize a global cost function, such as motion time, or to satisfy local objectives, such as passing an obstacle from the front rather than from the rear.

A trajectory consists of a sequence of avoidance maneuvers. A trajectory that minimizes motion time can be generated by searching over a tree of feasible avoidance maneuvers, generated at discrete time intervals. Figure 2 shows two branches of the tree, rooted in node $n_i$ at time $i$

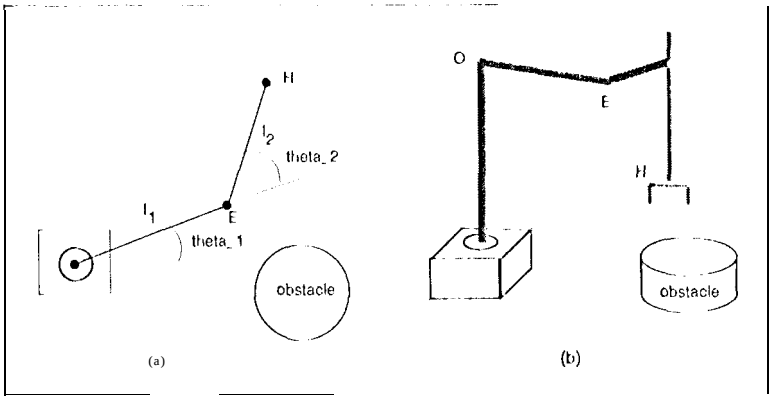


Figure 3: Planar 2-clef Manipulator: a) top view, b) side view

and reaching nodes $n_{i+1}$. The feasible avoidance velocities at times $i$ and $i$ -i 1 are represented by $\mathrm{RAV}^{i}$ and $\mathrm{RAV}^{i+1}$. A trajectory generated by this search is a good initial guess for the dynamic optimization, since it, is quasi optimal, and it has the desired topological properties (i.e. sequence of avoidance and type of maneuvers). A drawback of this trajectory is that its velocity profile is discon tinuous, and hence cannot be differentiated to compute the nominal controls. This is resolved by first smoothing the trajectory using Hermite splines, as discussed next.

### 3.2. Generating the Controls

'J'o compute the controls, we first smooth the trajectory, consisting of a sequence of avoidance maneuvers, using a spline interpolation. First, the path is smoothed by joining the mid-points of every consecutive path segments with a third order Hermite spline that matches the slopes of the path segments [14]. The velocity profile along the resulting path is smoothed using a cycloid between the mid-points of consecutive velocity segments, given by:

$$v(t) = \frac{\omega t - \sin(\omega t)}{2.0\pi} \tag{42}$$

where w = $\frac{2.0\pi}{T}$, and $T$ is the motion time between the two mid-points.

Using inverse dynamics, we now compute the controls associated with the smoothed trajectory. The switching times arc approximated at the zero crossings of the control signals, with a dead-band to avoid chatter.

## 4. Examples

Here we present examples for the two degree-of-freedom planar manipulator shown in Figure 3. The problem is greatly simplified by assuming a planar SCAR manipulator, with the end effector moving among obstacles placed below the plane of the links. The dynamic model of this manipulator is given in the Appendix C.

### 4.1. Single Obstacle

The objective in the following examples is to move the end-effector from rest at the starting position x = (- .15 $m$, .55 $m$), to rest at the goal position x = (1.5 $m$, - .5 $m$), in min imum time.

First, the optimal path, computed with no obstacles, is shown in Figure 4. The actuator torques for this solution are shown in Figure 5. For this case, the second joint has one switch, whereas the first joint has two switches and a possible singular arc (multiple switches) near the start point. This singular arc may be explained by the smaller angular rotation of the first joint compared to the rotation of the joint. This solution closely satisfies the necessary conditions of optimality, and is similar to the solution computed by the parameter optimization presented in [36]. The optimal time for this case is 3.59 $s$.

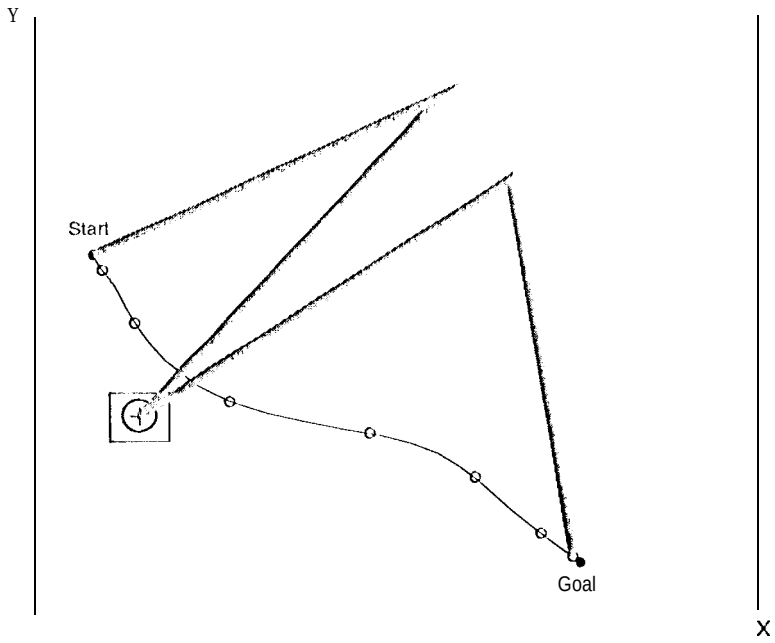


Figure 4: Optimal trajectory in the free environment

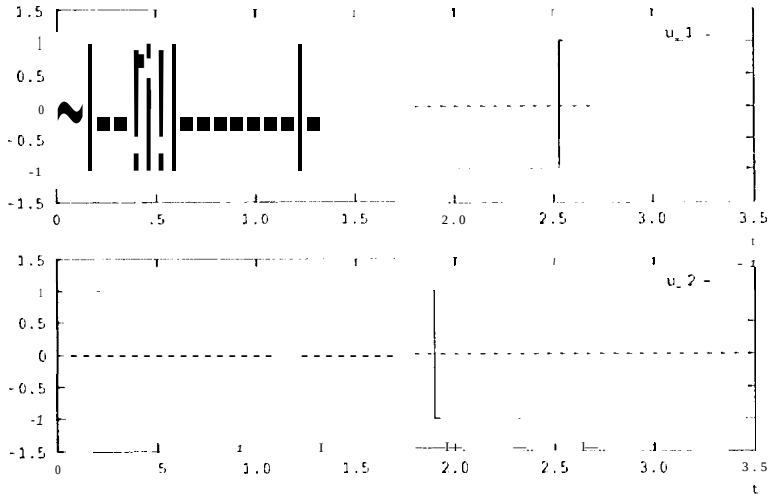


Figure 5: Optimal controls in the free environment

The second case considers a static obstacle, represented by a circle of radius $r = .4\ m$, centered at C = (.6 $m$, - .2 $m$). The constraints $\Psi_1$ and $\Psi_2$ due to this obstacle are:

$$\Psi_1 \ : \ \begin{pmatrix} (x - x_o)^2 + (y - y_o)^2 - r^2 = 0 \\ (x - x_o)v_x + (y - y_o)v_y = 0 \end{pmatrix} \quad t = t_1$$

$$\Psi_2 \ : \ v_x^2 + (x - x_o)a_x + v_y^2 + (y - y_o)a_y \geq 0 \quad t_1 \leq t \leq t_2$$

The optimal path for this case is shown in Figure 6, and the actuator torques are shown in Figure 7. Here the path grazes the obstacle at one point, and (lots not follow the obstacle because of its high curvature. The optimal time for this case is 5.1 7 $s$.

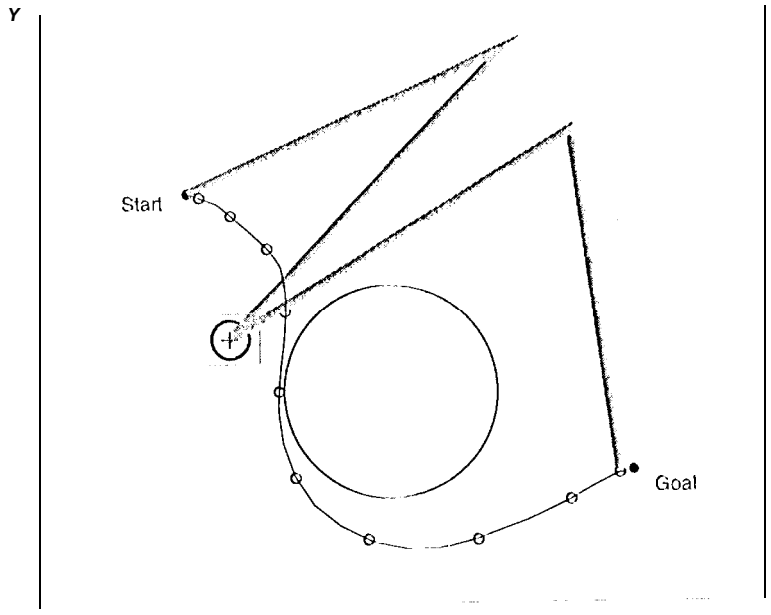


Figure 6: Optimal trajectory with a fixed obstacle

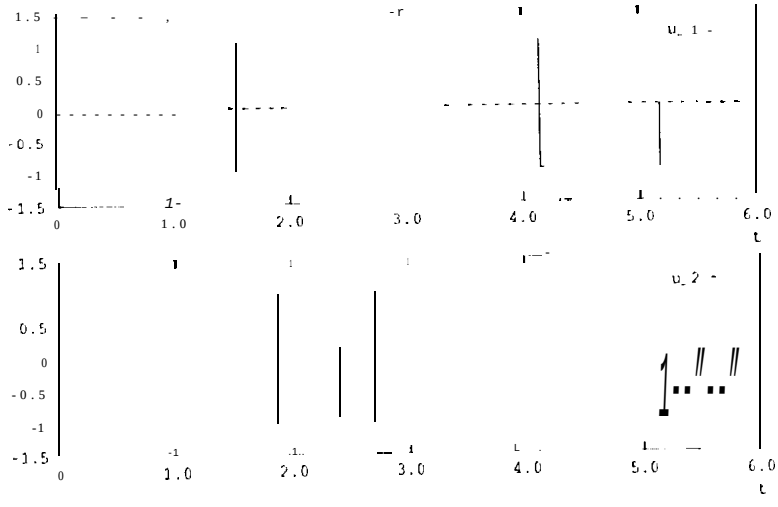


Figure 7: Optimal controls with a fixed obstacle

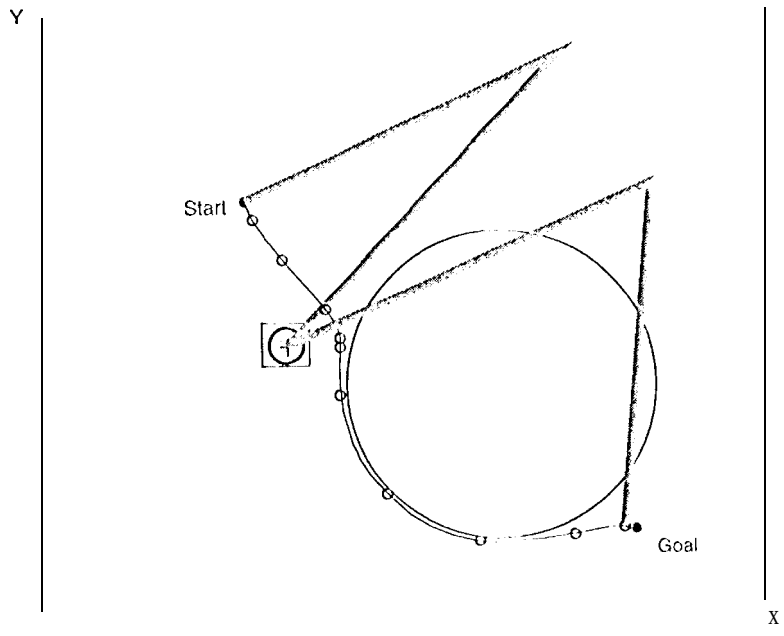


Figure 8: Optimal trajectory with a large static obstacle

This case was repeated with a larger obstacle, as shown in Figure 8, where the path follows the obstacle boundary. Here, the obstacle is of radius $r = .6\ m$, located at $\mathbf{C} = (.8\ m, -.15\ m)$.

Finally, the third **case** considers a moving obstacle, as shown in Figure 9. The constraints $\Psi_1$ and $\Psi_2$ are now:

$$\Psi_1 \; : \; \left( \begin{array}{rcl} (x - (v_{ox}t + x_o))^2 + (y - (v_{oy}t + y_o))^2 - r^2 & = & 0 \\ (x - (v_{ox}t + x_o))(v_x - v_{ox}) & & \\ + (y - (v_{oy}t + y_o))(v_y - v_{oy}) & = & 0 \end{array} \right) \quad t = t_1$$

$$\Psi_2 \; : \; \begin{array}{rcll} ((v_x - v_{ox})^2 + (x - (v_{ox}t + x_o))a_x + & & & \\ ((v_y - v_{oy})^2 + (y - (v_{oy}t + y_o))a_y & = & 0 & t_1 \leq t \leq t_2 \end{array}$$

The optimal path for this case, shown in Figure 9, slides along the moving obstacle. The actuator torques for this case are shown in Figure 10. The motion time for this case is $t = 4.36\,s$, which is longer than the unconstrained time, but shorter than the time with a fixed obstacle.

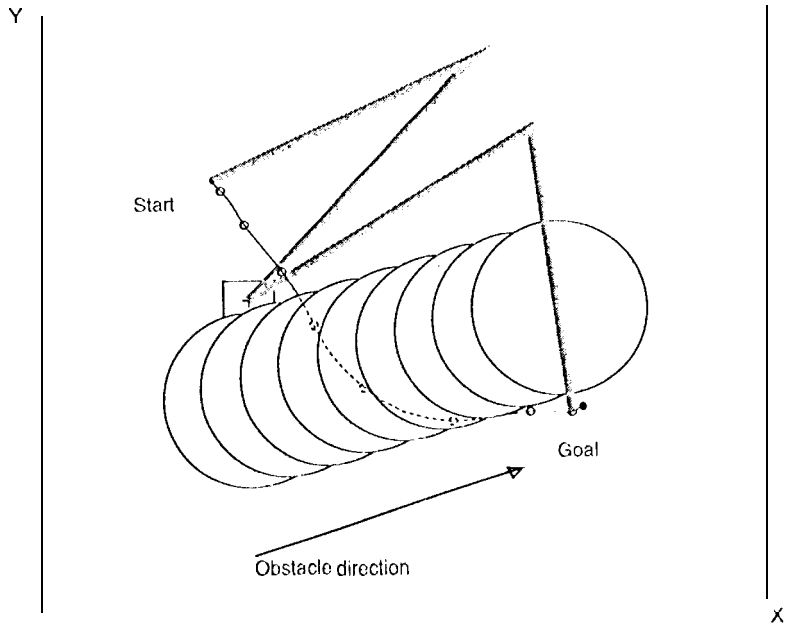


Figure 9: Optimal trajectory with a moving obstacle

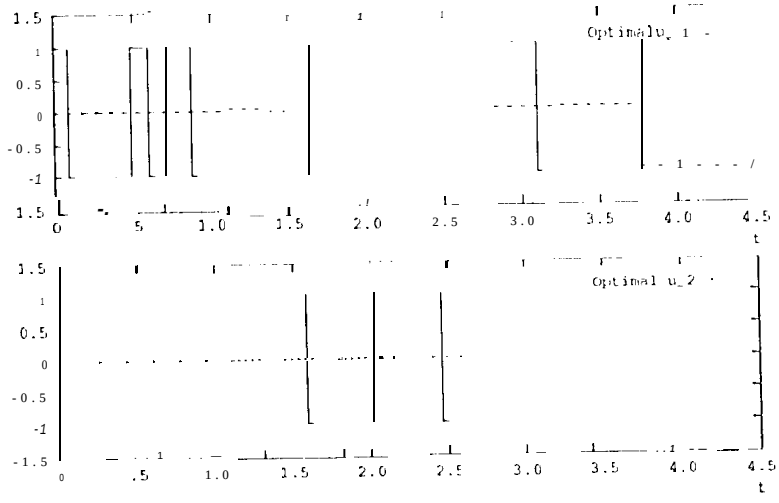


Figure 10: Optimal controls with a moving obstacle

### 4.2. Multiple obstacles

In this example, the optimal trajectory is computed for two moving obstacles, using the SCARA manipulator as in the previous examples. The obstacles are moving at constant velocities: obstacle 1 at (.045, .045) $m/s$ and obstacle 2 at (- .007, .03) $m/s$, starting at time $t_0$ from the positions (.1, - .5) $m$ and (1.15, .7) $m$, respectively. The end-effector starts at rest from (.3, ,2) $m$, and ends at rest at (1.5, - .5) $m$.

The optimal trajectory is computed by first generating an initial guess using a global search over a tree of avoidance maneuvers. The avoidance maneuvers were generated using velocity obstacles at $T$ = 1 s intervals. The motion time for the initial guess was 4.81 s. The bang-bang controls computed for this trajectory are shown in Figure 12. Optimizing from this initial guess resulted in the path shown in Figure 1 I, and the actuator torques shown in Figure 13. The optimal motion time for this case is 2.6 s. The improvement in motion time of the optimal trajectory compared to the initial guess is due to the fact that avoiding the velocity obstacles produces conservative trajectories, i .e. trajectories consisting of velocity segments that are guaranteed to avoid both obstacles at *all* times [1 1]. Clearly, the dynamic optimization is not subject to such a constraint, and therefore produces shorter motion times.

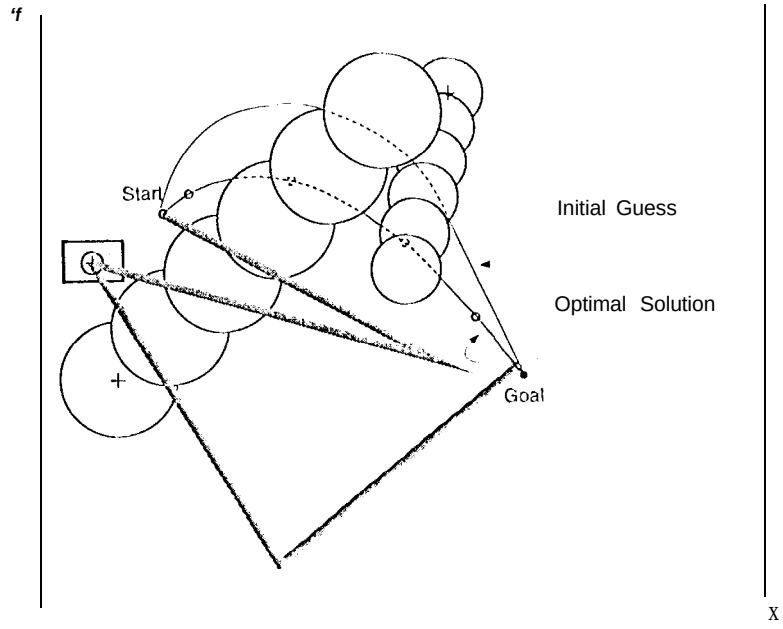


Figure 11: Optimal trajectory with two moving obstacles.

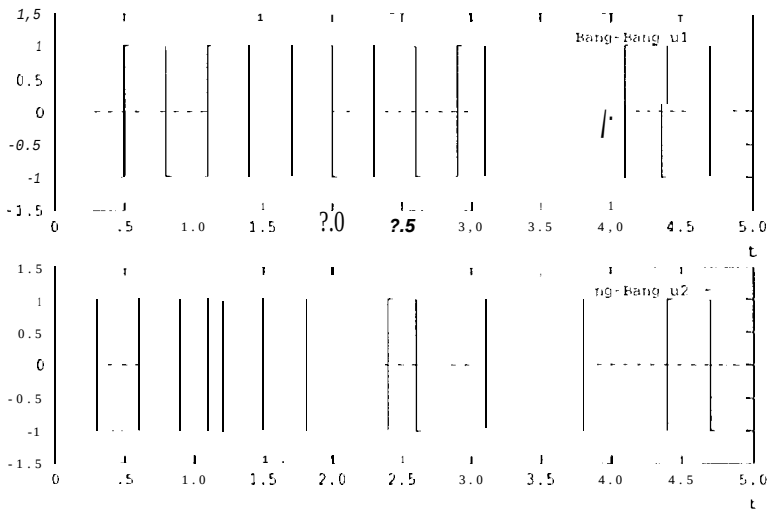


Figure 12: Bang-bang controls for the nominal trajectory.

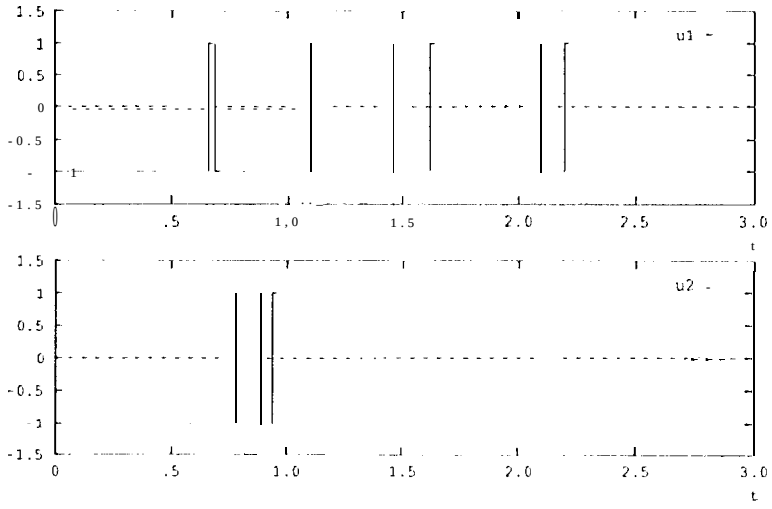


Figure 13: Bang-bang controls for the solution.

## 5. summary

This paper presented a method for computing the time-optimal trajectories of a manipulator moving in dynamic environments, subject to system dynamics and actuator constraints. Formulating the problem as a time-minimization, the state inequality constraints due to the moving obstacles are transformed to state-dependent control constraints and a tangency point constraint at the entry point of the constrained arc. Assuming bang-bang controls, this optimization problem is solved numerically as a parameter optimization over the switching times and final time, using a steepest descent algorithm. The initial guess for the optimization is computed using the previously developed concept of the *Velocity Obstacle* [10]. The velocity obstacles allow one to select an initial guess that has a desirable structure, i.e. a desirable sequence of avoidance and a desirable side from which each obstacle should be avoided. The method is demonstrated in several examples for a 2-DOF planar manipulator moving amongst static and moving circular obstacles.

The optimal trajectory in a free environment compares favorably with the solution computed by another method [36], thus verifying the correctness of the proposed approach. Clearly, this method is meant for off-line computations, and is thus applicable to repetitive tasks, such as manipulators operating between moving conveyor belts, or manipulators operating off moving platforms. A more efficient method for on-line planning (with no guarantee of optimality) in dynamic environments has been presented in [13].

## 6. Acknowledgment

The research described in this paper has been partially carried out at the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

## 1 REFERENCES


1. H Asada and J.-J. Slotine. *Robot A nalysis and Control.* John Wiley and Sons, New York, 1986.
2. J.E. Bobrow. Optimal robot path planning using the minimum time criterion. *IEEE Transaction on Robotics and Automation*, 4(4):443 450, 1988.
3. A .E. Bryson. *Dynamic Optimization.* Stanford University, 1 'alo Alto, CA, March *231992.*
4. A. It. Bryson. Inverse Dynamic Optimization. MANE Seminar, UCLA, 1995.
5. A.E. Bryson and S.E. Denham, W.F.and Dreyfus. Optimal programming problems with inequality constraints i: Necessary conditions for extremal solutions. *A IA A Journal, 1 (I 1*):2544-2550, November 1963.
6. A.E. Bryson and W.F. Denham. A steepest- ascent method for solving optimum programming problems. *A SME Journal of Applied Mechanics*, (29):247- *257,* June 1962.
7. A.E. Bryson and Y.C. Ho. *Applied* **Optimal** *Control.* Hemisphere Publishing Corp., New York, NY, 1975.
8. A .E. Denham, W. F.and Bryson. Optimal programming problems with inequality constraints *ii*: Solution by steepest ascent. *AIA A Journal*, 2(2):25- *34,* January 1964.
9. M. Erdmann and 'J'. Lozano-Perez. On multiple moving objects. *A lgorithmica*, (2):477- *521, 1987.*
10. *1 '*. Fiorini. *Robot Motion Planning among Moving Obstacles.* PhD thesis, University of California, Los Angeles, January 1995.
11. 1'. Fiorini and Z. Shiller. Motion planning in dynamic environments using the relative velocity paradigm. In *IEEE Internation* al *Conference of* A *utomation and Robotics*, volume 1, pages 560-566, May 1993.
12. 1'. Fiorini and Z. Shiller. Robot motion planning in dynamic environments. In G. Girald and G. Hirzinger, editors, *International Symposium of Robotic Research*, pages 237-248, Munich, Germany, 20-24 October 1995. Springer-Verlag.
13. P. Fiorini and Z. Shiller. Time optimal trajectory planning in dynamic environments. In *IEEE International Conference of A utomation and Robotics,* volume 2, pages 1553 1 558, Minneapolis, MN, 22-28 April 1996.
14. J.D. Foley, A. van Dam, S.K. Feiner, and J.F. Hughes. *Computer Graphics.* A ddison - Wesley Publishing Company, Reading, MA, 1990.
15. ' J 's Fraichard. Dynamic trajectory planning with dynamic constraints: a state-time space approa ch. In *IEEE/RSJ Intern ational Conference on Intelligent Robots a nd Systems*, pages 1393 1400, Yokohama, Japan, 26-30 July 1993.
16. T. Fraichard and C Laugier. Path-velocity decomposition revisited and applied to dynamic trajectory planning. in *IEEE International Conference of A utomation and Robotics*, vol - ume 1, pages *40* 45, Atlanta, GA, 2-6 May 1993.
17. K. Fujimura. Motion planning amid transient obstacles. *International Journal of Robotics Research, 1* 3(5):395- 407, October 1994.
18. K. Fujimura. Time-minimum routes in time-dependent networks. *IEEE Transa ction on Robotics and A utomation*, 11(3):343 351, June 1995.
19. *K.* Fujimura and 11. Samet. A hierarchical strategy for path planning among moving obstacles. *IEEE Transaction on Robotics and A utomation*, 5(1):61 -*69,* February 1989.
20. K. Fujimura and H. Samet. Time-minimal paths among moving obstacles. In *IEEE International Conference on Robotics and A utomation*, pages 1 1 10- ] ] 15, Scottsdale, AZ, May 1989.
21. K. Fujimura and H. Samet. Motion planning in a dynamic domain. In *IEEE International*

*Conference on Robotics and Automation*, pages 324-330, Cincinnati, 011, May 1990.

22. D.H. Jacobson, M.M. Lele, and J.L. Speyer. New necessary conditions of optimality for control problems with stale!-v;irial)lc inequality constraints. *Journal of Mathematical Analysis and Applications*, 35(2):255-*284*, 1971.
23. 1 ).W. Johnson and E. G. Gilbert. Minimum time robot planning in the presence of obstacles. 111 *IEEE Conference on Decision and Control*, pages 1748 1 753, Ft. Lauderdale, Fl., December 1985.
24. M.E. Kahn. *The Near-Minimum Time Control of Open Loop Articulated Kinematic Chains.* PhD thesis, Stanford, 1969.
25. ]{. Kant and S.W. Zucker. Towards efficient trajectory planning: the lmtl]-velocity decomposition. *The International Journal of Robotic Research*, 5(3) :*72-89*, Fall 1986.
26. J .C. 1 atombe, editor. *Robot Motion Planning.* Kluwer Academic Publisher, Boston, MA, ] 00].
27. J.]'. Laumond. Feasible trajectories for mobile robots with kinematic and environment constraints. In 1,.(). Hertzberger and F.C.A. Groen, editors, *Intelligent Autonomous Systems*, pages *346-354*, New York, NY, 1987. North Holland.
28. J .1'. Laumond, P.E. Jacobs, M. q'six, and Murray R.M. Motion planner for non-holonomic mobile robots. *IEEE Transactions on Robotics and Automation*, RA-10(5):573-593, 1 994.
29. B.H. Lee and C.S. G. Lee. Collision-free motion planning of two robots. *IEEE Transactions on System Man and Cybernetics*, SMC-1 7(1):21-32, January-February 1987.
30. T. Lozano-Pérez and M. A. Wesley. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22(1 0):560 570, October 1979.
31. E.B. Meier. *An Efficient Algorithm for Bang-Bang Control Systems Applied to a Two-Link Manipulator.* P])]) thesis, Stanford, December 1987.
32. R.M. Murray and S.S. Sastry. Nonholonomic motion planning: steering with sinusoids. *IEEE Transactions on Automatic Control*, AC-38:700 716, 1 993.
33. J. Reif and M. Sharir. Motion planning in the presence of moving obstacles. In *25th IEEE Symposium on the Foundation* oj *Computer Science*, pages 144-153, 1 985.
34. H. Seywald. Trajectory optimization basal on differential inclusion. *AIA A Journal of Guidance, Control and Dynamics*, 17(3), 1994 .
35. Z. Shiller. Time-energy optimal control of articulated systems with geometric path constraints. *ASME Jo urnal of Dynamic Systems, Measurements and Control*, 11 8(1 ):139 *143*, March 1996.
36. Z. Shiller and S. Dubowsky. Robot path planner with obstacles, actuators, gripper and payload constraints. *The International Journal of Robotics Research*, 8(6):3-18, December *1989.*
37. Z. Shiller and R.Y. Gwo. 1 Dynamic motion planning of autonomous vehicles. *IEEE Transactions of Robotics and Automation*, 7(2):241-249, April 1 991.
38. Z. Shiller and S. Sundar. Emergency maneuvers of autonomous vehicles. In *The 13th World Congress, IFA C'96*, volume Q, pages 393 398, San Francisco, CA, July 1996.
39. J. Speyer. *Optimization and Control oj Nonlinar Systems with Inflight Constraints.* PhD thesis, Harvard, February 1968.
40. A. Weinreb and A .E. Bryson. Optimal control of systems with hard control bounds. *IEEE Transactions on Automatic Control*, AC-30(6), November 1985.
41. Tsuneo Yoshikawa. *Foundation of Robotics.* The MIT Press, Cambridge, MA, 1990.

# Appendix

## A. Derivation of the Terminal Differential

The differential of the terminal constraint $\Omega$ can be computed using [6]:

$$(d\Omega)_{t_f} = \left(\frac{\partial \Omega}{\partial x} dx + \frac{\partial \Omega}{\partial t} dt\right)_{t_f} \tag{43}$$

Using $dx = \delta x + \dot{x} dt$ it follows that

$$(d\Omega)_{t_f} = (\delta\Omega)_{t_f} + \dot{\Omega}_{t_f} dt_f \tag{44}$$

The variation $\delta x$ satisfies the first order perturbation equation:

$$\delta \dot{x} = \frac{\partial \mathcal{F}}{\partial x} \delta x + \frac{\partial \mathcal{F}}{\partial u} \delta u \tag{45}$$

Therefore, there exists a state transition matrix $\Phi(t, \tau)$ expressing the variation $(\delta x)_{tf}$ [7]. The variation $\delta\Omega$ is then:

$$(\delta\Omega)_{t_f} = \left.\frac{\partial \Omega}{\partial x}\right|_{t_f} \left(\Phi(t_f, t_0)\delta x(t_0) + \int_{t_0}^{t_f} \Phi(t_f, \tau) \frac{\partial \mathcal{F}}{\partial u} \delta u(\tau) d\tau\right) \tag{46}$$

This expression can be simplified by defining a multiplier $\lambda_\Omega \in \Re^n \times \Re^l$ as:

$$\lambda_\Omega^T(t) = \left(\frac{\partial \Omega}{\partial x}\right)_{t_f} \Phi(t_f, t) \tag{4'7}$$

where $n$ is the dimension of $\mathbf{x}$ and $l$ is the number of terminal constraints. Taking advantage of the properties of the state transition matrix $\Phi$ [7], a set of adjoint equations for $\lambda_\Omega$ can be written as:

$$\dot{\lambda}_\Omega(t) = -\left(\frac{\partial \mathcal{F}}{\partial x}\right)^T \lambda_\Omega(t) \qquad \lambda_\Omega(t_f) = \left(\frac{\partial \Omega}{\partial x}\right)_{t_f} \tag{49}$$

(48)

Therefore, using

$$(\delta\Omega)_{t_f} = \int_{t_0}^{t_f} \lambda_\Omega^T \frac{\partial \mathcal{F}}{\partial \mathbf{u}} \delta \mathbf{u} d\tau + \delta\Omega_0 \tag{50}$$

in equation *(44),* and assuming fixed initial conditions, the total differential of $\Omega$ becomes:

$$d\Omega(t_f) = \int_{t_0}^{t_f} \lambda_\Omega^T \frac{\partial \mathcal{F}}{\partial \mathbf{u}} \delta \mathbf{u} d\tau + \left(\frac{\partial \Omega}{\partial x} \dot{x} + \frac{\partial \Omega}{\partial t}\right)_{t_f} dt_f \tag{51}$$

### B. Effect of the Point Constraint on the Multipliers

The co-state equations for $\lambda_{phi}$, $\lambda_\Omega$ used in the previous Sections do not take into account the effects of the constraints $\Psi_1$ and $\Psi_2$ given by:

$$\boldsymbol{\Lambda}^T_{t_1^-} = \boldsymbol{\Lambda}^T_{t_1^+} + \eta^T \frac{\partial \Psi_1}{\partial x(t_1)} \tag{52}$$

This discontinuity affects the co-state equations, as illustrated in the following using multipliers $\lambda_\Omega$ [8].

The unknown $\eta$ is computed by relating the value of $\lambda_\Omega$ at $t_1^-$, i.e. just before reaching the constraint $\Psi_1$, to its value at $t_1^+$, i.e. just after reaching $\Psi_1$. To do this, $\lambda_\Omega(t_1^+)$ and $\lambda_\Omega(t_1^-)$ are first computed independently of each other, using the expressions for $d\Omega$ at $t_f$ and $t_1$.

The value of $\lambda_\Omega(t_1^+)$ is computed from the expression of the changes in $d\Omega(t_f)$ due to the variation of $\mathbf{x}$, $\delta\mathbf{x}(t_1^+)$:

$$d\Omega(t_f) = \lambda^T_\Omega \delta\mathbf{x}(t_1^+) = \lambda^T_\Omega (d\mathbf{x} - \dot{\mathbf{x}} dt_1)_{t_1^+} \tag{53}$$

that can be rewritten as:

$$d\Omega(t_f) = \left(\frac{\partial \Omega}{\partial \mathbf{x}} d\mathbf{x} + \frac{\partial \Omega}{\partial t} dt\right)_{t_1^+} \tag{54}$$

from which:

$$\frac{\partial \Omega}{\partial \mathbf{x}_1} = \lambda^T_\Omega(t_1^+) \tag{55}$$

$$\frac{\partial \Omega}{\partial t_1} = -\lambda^T_\Omega(t_1^+)\dot{\mathbf{x}}(t_1^+) \tag{56}$$

The value of $d\Omega$ at $t_1^-$ is computed using:

$$d\Omega(t_1^-) = (\delta\Omega)_{t_1^-} + \dot{\Omega}_{t_1^-} dt_1^- \tag{57}$$

where:

$$(\delta\Omega)_{t_1^-} = (\lambda_\Omega \delta\mathbf{x})_{t_0} + \int_{t_0}^{t_1^-} \lambda^T_\Omega \frac{\partial \mathcal{F}}{\partial \mathbf{u}} \delta\mathbf{u} d\tau$$

Since $dS^{(p-1)}(\mathbf{x}) = 0$, the value of $dt_1$ is:

$$dt_1 = -\frac{1}{\dot{S}^{(p-1)}} \left[-\left(\lambda^T_{S^{(p-1)}} \delta\mathbf{x}\right)_{t_0} - \int_{t_0}^{t_1} \lambda^T_{S^{(p-1)}} \frac{\partial \mathcal{F}}{\partial \mathbf{u}} \delta\mathbf{u} dt\right] \tag{58}$$

By replacing $dt_1$ in $d\Omega$ of equation (57) with (58), and since $S^{(p-1)}$ and $\dot{\Omega}$ are both independent of the integration variable, $d\Omega(t_1^-)$ becomes:

$$d\Omega(t_1^-) = \left(\lambda^T_\Omega \delta\mathbf{x}\right)_{t_0} + \int_{t_0}^{t_1^-} \left(\lambda^T_\Omega - \frac{\dot{\Omega}}{\dot{S}^{(p-1)}} \lambda^T_{S^{(p-1)}}\right) \frac{\partial \mathcal{F}}{\partial \mathbf{u}} \delta\mathbf{u} d\tau - \left.\frac{\dot{\Omega}}{\dot{S}^{(p-1)}}\right|_{t_1^-} \left(\lambda^T_{S^{(p-1)}} \delta\mathbf{x}\right)_{t_0} \tag{59}$$

The desired expression of $\lambda_\Omega$ at $t_1^-$, satisfying $d(S^{(p-1)}) = 0$ is then:

$$\lambda^T_{\Omega, S^{(p-1)}}(t_1^-) = \left(\lambda^T_\Omega - \frac{\dot{\Omega}}{\dot{S}^{(p-1)}} \lambda^T_{S^{(p-1)}}\right)_{t_1^-} \tag{60}$$

This equation can be further simplified by replacing $\dot{\Omega}(t_1^-)$ with:

$$\dot{\Omega}(t_1^-) = \left.\frac{\partial \Omega}{\partial \mathbf{x}}\right|_{t_1^-} \dot{\mathbf{x}}(t_1^-) - \lambda_\Omega^T(t_1^+)\dot{\mathbf{x}}(t_1^+) \tag{61}$$

Since the differentials $d\mathbf{x}(t_1)$ and $dt_1$ are the same at $t_1^-$ and $t_1^+$, equation (55) gives:

$$\left.\frac{\partial \Omega}{\partial \mathbf{x}}\right|_{t_1^-} = \lambda_\Omega^T(t_1^+) \tag{62}$$

and similarly

$$\lambda^T_{S^{(p-1)}}(t_1^-) = \left.\frac{\partial S^{(p-1)}}{\partial \mathbf{x}}\right|_{t_1} \tag{63}$$

By using equations (62), (63), and (61) in (60), the discontinuity of $\lambda_\Omega$ at, $t_1$ becomes:

$$\lambda^T_{\Omega,S^{(p-1)}}(t_1^-) = \lambda_\Omega^T(t_1^+)\left(I - \frac{\dot{\mathbf{x}}(t_1^-) - \dot{\mathbf{x}}(t_1^+)}{S^{(p)}(t_1)} \left.\frac{\partial S^{(p-1)}}{\partial \mathbf{x}}\right|_{t_1}\right) \tag{64}$$

which is equivalent to the necessary condition (52) if the multiplier $\eta$ is equal to:

$$\eta^T = -\lambda_\Omega^T(t_1^+)\left(\frac{\dot{\mathbf{x}}(t_1^-) - \dot{\mathbf{x}}(t_1^+)}{S^{(p)}(t_1^-)}\right) \tag{65}$$

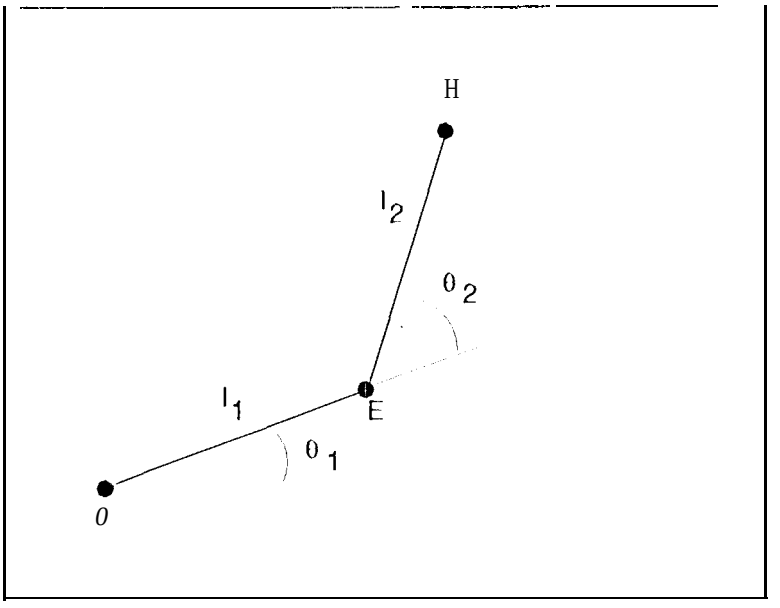


Figure 14: Planar 2-dof Manipulator

### C. Kinematic and Dynamic Equations of a Two-Link Manipulator

Figure 14 shows the model of the two link manipulator used in the previous examples. The symbols used in the model are the following:

o the shoulder of the arm,

E the elbow of the arm,

H the tip of the forearm,

$C_1$ center of mass of link one,

$C_2$ center of mass of link two,

$l_1$ length of link one,

$l_2$ length of link two,

$l_{C_1}$ distance of $C_1$ from O,

$l_{C_2}$ distance of $C_2$ from A,

$m_1$ mass of link one,

$m_2$ mass of link two,

$\theta_1$ angle between link one and the X axis,

$\theta_2$ angle between link two and link one,

I principal central moment of inertia of each link, $I = \frac{m l_i^2}{12}$

$\tau_{1,2}$ torques applied at joints 1 and 2.

The kinematic equations of the arm are [41]:

- Direct kinematics:

$$x_h = l_1 \cos(\theta_1) + l_2 \cos(\theta_1 + \theta_2) \quad (66)$$
$$y_h = l_1 \sin(\theta_1) + l_2 \sin(\theta_1 + \theta_2) \quad (67)$$

- Inverse kinematics:

$$\theta_1 = \arctan 2(y_h, x_h) \mp \arctan 2(k, x_h^2 + y_h^2 + l_1^2 - l_2^2) \quad (68)$$
$$\theta_2 = \pm \arctan 2(k, x_h^2 + y_h^2 - l_1^2 - l_2^2) \quad (69)$$
$$k = \sqrt{(x_h^2 + y_h^2 + l_1^2 + l_2^2)^2 - 2\,[(x_h^2 + y_h^2)^2 + l_1^4 + l_2^4]} \quad (70)$$
$$(71)$$

with the condition

$$(l_1^2 - l_2^2) \leq (x_h^2 + y_h^2) \leq (l_1^2 + l_2^2)$$

In the symbols $(\mp, \pm)$, the top sign refers to the *elbow up* configuration of the arm, and the bottom sign refers to the *elbow down* configuration.

. Differential kinematics:

$$J = \begin{bmatrix} -l_1 \sin\theta_1 - l_2 \sin(\theta_1 + \theta_2) & -l_2 \sin(\theta_1 + \theta_2) \\ l_1 \cos\theta_1 + l_2 \cos(\theta_1 + \theta_2) & l_2 \cos(\theta_1 + \theta_2) \end{bmatrix} \quad (72)$$

$$J^{-1} = \begin{bmatrix} \frac{\cos(\theta_1+\theta_2)}{l_1 \sin\theta_2} & \frac{\sin(\theta_1+\theta_2)}{l_1 \sin\theta_2} \\ -\frac{l_1 \cos\theta_1 + l_2 \cos(\theta_1+\theta_2)}{l_1 l_2 \sin\theta_2} & -\frac{l_1 \sin\theta_1 + l_2 \sin(\theta_1+\theta_2)}{l_1 l_2 \sin\theta_2} \end{bmatrix} \quad (73)$$

The dynamic equations of the arm are [1]:

- State equations

$$x_1 = \theta_1 \quad (74)$$
$$x_2 = \dot{\theta}_1 \quad (75)$$
$$x_3 = \theta_2 \quad (76)$$
$$x_4 = \dot{\theta}_2 \quad (77)$$

- Dynamic equations

$$\dot{x_1} = x_2 \quad (78)$$
$$\dot{x_2} = (\tau_1 - H_{12}\dot{x_4} + h x_4^2 + 2h\, x_2 x_4)/H_{11} \quad (79)$$
$$\dot{x_3} = x_4 \quad (80)$$
$$\dot{x_4} = (\tau_2 - H_{12}\dot{x_2} - h x_2^2)/H_{22} \quad (81)$$

with:

$$H_{11} = m_1 l_{c_1}^2 + I_1 + m_2(l_1^2 + l_{c_2}^2 + 2 l_1 l_{c_2} \cos\theta_2) + I_2 \quad (82)$$

$$H_{22} = m_2 l_{c_2}^2 + I_2 \quad (83)$$

$$H_{12} = m_2 l_1 l_{c_2} \cos\theta_2 + m_2 l_{c_2}^2 + I_2 \quad (84)$$

$$h = m_2 l_1 l_{c_2} \sin\theta_2 \quad (85)$$

From these equations it follows that:

$$\dot{x}_2 = (\tau_1 - \frac{H_{12}}{H_{22}}\tau_2 + \frac{h H_{12}}{H_{22}} x_2^2 + h x_4^2 + 2 h x_2 x_4)\frac{H_{22}}{H_{11} H_{22} - H_{12}^2} \quad (86)$$

$$\dot{x}_4 = (\tau_2 - H_{12}\dot{x}_2 - h x_2^2)/H_{22} \quad (87)$$

The values of the parameters used in the examples are the following:

- $l_1 = 1.5111$, $l_2 = 1.3111$.
- $m_1 = 10.0$ Kg, $m_2 = 10.0$ Kg.
- $\tau_1 = 10.0$ Nm, $\tau_2 = 3.0$ Nm.